```
from google.colab import files
f=files.upload()
```

Choose Files AirQuality.csv

- **AirQuality.csv**(text/csv) - 62540857 bytes, last modified: 3/3/2022 - 70% done

```
import pandas as pd
df=pd.read_csv(url,encoding='cp1252')
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWarning: Columns (0) have mixed types.Specify dtype c
  interactivity=interactivity, compiler=compiler, result=result)
```

```
df.head()
```

	stn_code	sampling_date	state	location	agency	type	so2	no2	rspm	spm
0	150	February - M021990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	4.8	17.4	NaN	NaN
1	151	February - M021990	Andhra Pradesh	Hyderabad	NaN	Industrial Area	3.1	7.0	NaN	NaN
2	152	February -	Andhra	Hyderabad	NaN	Residential, Rural and	6.2	28.5	NaN	NaN

```
df.shape
```

```
(435742, 13)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435742 entries, 0 to 435741
Data columns (total 13 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   stn_code                     291665 non-null  object
 1   sampling_date                435739 non-null  object
 2   state                        435742 non-null  object
 3   location                     435739 non-null  object
 4   agency                       286261 non-null  object
 5   type                         430349 non-null  object
 6   so2                          401096 non-null  float64
 7   no2                          419509 non-null  float64
 8   rspm                         395520 non-null  float64
 9   spm                          198355 non-null  float64
 10  location_monitoring_station  408251 non-null  object
 11  pm2_5                        9314 non-null    float64
 12  date                         435735 non-null  object
dtypes: float64(5), object(8)
memory usage: 43.2+ MB
```

```
df.isnull().sum()
```

```
stn_code                       144077
sampling_date                       3
state                               0
location                            3
agency                         149481
type                             5393
so2                             34646
no2                             16233
rspm                            40222
spm                            237387
location_monitoring_station     27491
pm2_5                          426428
date                                7
dtype: int64
```

```
df.count()   #It results in a number of non null values in each column.
```

```
stn_code                       291665
sampling_date                  435739
```

```
state                          435742
location                       435739
agency                         286261
type                           430349
so2                            401096
no2                            419509
rspm                           395520
spm                            198355
location_monitoring_station    408251
pm2_5                            9314
date                           435735
dtype: int64
```

Summarised details

Generate descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset’s distribution, excluding NaN values.

```
df.describe()
```

	so2	no2	rspm	spm	pm
count	401096.000000	419509.000000	395520.000000	198355.000000	9314.000
mean	10.829414	25.809623	108.832784	220.783480	40.791
std	11.177187	18.503086	74.872430	151.395457	30.832
min	0.000000	0.000000	0.000000	0.000000	3.000
25%	5.000000	14.000000	56.000000	111.000000	24.000
50%	8.000000	22.000000	90.000000	187.000000	32.000
75%	13.700000	32.200000	142.000000	296.000000	46.000

Cleansing the dataset

**In this step, we need to clean the data by adding and dropping the needed and unwanted data respectively. **

From the above dataset,

Dropping of less valued columns: stn_code, agency, sampling_date, location_monitoring_agency do not add much value to the dataset in terms of information. Therefore, we can drop those columns.

Changing the types to uniform format: When you see the dataset, you may notice that the ‘type’ column has values such as ‘Industrial Area’ and ‘Industrial Areas’ — both actually mean the same, so let’s remove such type of stuff and make it uniform.

Creating a year column To view the trend over a period of time, we need year values for each row and also when you see in most of the values in date column only has ‘year’ value. So, let’s create a new column holding year values.

1.stn_code, agency, sampling_date, location_monitoring_agency do not add much value to the dataset in terms of information. Therefore, we can drop those columns.

2.Dropping rows where no date is available.

```
df=df.drop(['stn_code', 'agency','sampling_date','location_monitoring_station'], axis = 1)  #dropping columns that aren't required
```

```
df=df.dropna(subset=['date']) # dropping rows where no date is available
```

```
df.columns
```

```
Index(['state', 'location', 'type', 'so2', 'no2', 'rspm', 'spm', 'pm2_5',
       'date'],
      dtype='object')
```

Changing the types to uniform format:

Notice that the ‘type’ column has values such as ‘Industrial Area’ and ‘Industrial Areas’—both actually mean the same, so let’s remove them and make it uniform

```
df["type"].unique()
```

```
array(['RRO', 'I', nan, 'S', 'RO', 'R', 'RIRUO'], dtype=object)
```

```
types = {
    "Residential": "R",
    "Residential and others": "RO",
    "Residential, Rural and other Areas": "RRO",
    "Industrial Area": "I",
    "Industrial Areas": "I",
    "Industrial": "I",
    "Sensitive Area": "S",
    "Sensitive Areas": "S",
    "Sensitive": "S",
    "NaN": "RRO"
}
df.type = df.type.replace(types)
```

```
df.head(5)
```

	state	location	type	so2	no2	rspm	spm	pm2_5	date
0	Andhra Pradesh	Hyderabad	RRO	4.8	17.4	NaN	NaN	NaN	1990-02-01
1	Andhra Pradesh	Hyderabad	I	3.1	7.0	NaN	NaN	NaN	1990-02-01
2	Andhra Pradesh	Hyderabad	RRO	6.2	28.5	NaN	NaN	NaN	1990-02-01

Creating a year column

To view the trend over a period of time, we need year values for each row and also when you see in most of the values in date column only has ‘year’ value. So, let’s create a new column holding year values.

```
df['date'] = pd.to_datetime(df['date'], errors='coerce')
df.head(5)
```

	state	location	type	so2	no2	rspm	spm	pm2_5	date
0	Andhra Pradesh	Hyderabad	RRO	4.8	17.4	NaN	NaN	NaN	1990-02-01
1	Andhra Pradesh	Hyderabad	I	3.1	7.0	NaN	NaN	NaN	1990-02-01
2	Andhra Pradesh	Hyderabad	RRO	6.2	28.5	NaN	NaN	NaN	1990-02-01

```
df['year'] = df.date.dt.year
df.head(5)
```

	state	location	type	so2	no2	rspm	spm	pm2_5	date	year
0	Andhra Pradesh	Hyderabad	RRO	4.8	17.4	NaN	NaN	NaN	1990-02-01	1990
1	Andhra Pradesh	Hyderabad	I	3.1	7.0	NaN	NaN	NaN	1990-02-01	1990
2	Andhra Pradesh	Hyderabad	RRO	6.2	28.5	NaN	NaN	NaN	1990-02-01	1990

Handling Missing Values

The column such as SO2, NO2, rspm, spm, pm2_5 are the ones which contribute much to our analysis. So, we need to remove null from those columns to avoid inaccuracy in the prediction. We use the Imputer from sklearn.preprocessing to fill the missing values in every column with the mean.

```
# defining columns of importance, which shall be used reguarly
COLS = ['so2', 'no2', 'rspm', 'spm', 'pm2_5']
```

```
import numpy as np
from sklearn.impute import SimpleImputer
# invoking SimpleImputer to fill missing values
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
df[COLS] = imputer.fit_transform(df[COLS])
```

```
df.head(5)
```

	state	location	type	so2	no2	rspm	spm	pm2_5	da
0	Andhra Pradesh	Hyderabad	RRO	4.8	17.4	108.833091	220.78348	40.791467	19 02
1	Andhra Pradesh	Hyderabad	I	3.1	7.0	108.833091	220.78348	40.791467	19 02
2	Andhra	Hyderabad	RRO	6.2	28.5	108.833091	220.78348	40.791467	19

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 435735 entries, 0 to 435738
Data columns (total 10 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   state     435735 non-null  object
 1   location  435735 non-null  object
 2   type      430345 non-null  object
 3   so2       435735 non-null  float64
 4   no2       435735 non-null  float64
 5   rspm      435735 non-null  float64
 6   spm       435735 non-null  float64
 7   pm2_5     435735 non-null  float64
 8   date      435735 non-null  datetime64[ns]
 9   year      435735 non-null  int64
dtypes: datetime64[ns](1), float64(5), int64(1), object(3)
memory usage: 36.6+ MB
```

Data Transformation

All machine learning algorithms are based on mathematics. So, we need to convert all the columns into numerical format.

Taking a broader perspective, data is classified into numerical and categorical data:

1. Numerical: As the name suggests, this is numeric data that is quantifiable.

2. Categorical: The data is a string or non-numeric data that is qualitative in nature.

---

1. Encoding To address the problems associated with categorical data, we can use encoding. This is the process by which we convert a categorical variable into a numerical form. Here, we will look at three simple methods of encoding categorical data.
2. Replacing This is a technique in which we replace the categorical data with a number. This is a simple replacement and does not involve much logical processing. Let's look at an exercise to get a better idea of this.

Simple Replacement of Categorical Data with a Number

```
df.head(5)
```

	state	location	type	so2	no2	rspm	spm	pm2_5	da
0	Andhra Pradesh	Hyderabad	RRO	4.8	17.4	108.833091	220.78348	40.791467	19 02
1	Andhra Pradesh	Hyderabad	I	3.1	7.0	108.833091	220.78348	40.791467	19 02
2	Andhra	Hyderabad	RRO	6.2	28.5	108.833091	220.78348	40.791467	19

```
df['type'].value_counts()
```

```
RRO      179013
I        148069
RO        86791
S         15010
RIRUO      1304
R           158
Name: type, dtype: int64
```

```
df['type'].replace({"RRO":1, "I":2, "RO":3,"S":4,"RIRUO":5,"R":6}, inplace= True)
```

```
df['type']
```

```
0         1.0
1         2.0
2         1.0
3         1.0
4         2.0
         ...
435734    5.0
435735    5.0
435736    5.0
435737    5.0
435738    5.0
Name: type, Length: 435735, dtype: float64
```

Converting Categorical Data to Numerical Data Using Label Encoding

> Indented block

```
df['state'].value_counts()
```

```
Maharashtra             60382
Uttar Pradesh           42816
Andhra Pradesh          26368
Punjab                  25634
Rajasthan               25589
Kerala                  24728
Himachal Pradesh        22896
West Bengal             22463
Gujarat                 21279
Tamil Nadu              20597
Madhya Pradesh          19920
Assam                   19361
Odisha                  19278
Karnataka               17118
Delhi                    8551
Chandigarh               8520
Chhattisgarh             7831
Goa                      6206
Jharkhand                5968
Mizoram                  5338
Telangana                3978
Meghalaya                3853
Puducherry               3785
Haryana                  3420
Nagaland                 2463
Bihar                    2275
Uttarakhand              1961
```

```
Jammu & Kashmir          1289
Daman & Diu               782
Dadra & Nagar Haveli      634
Uttaranchal               285
Arunachal Pradesh          90
Manipur                    76
Sikkim                      1
Name: state, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
df["state"]=labelencoder.fit_transform(df["state"])
df.head(5)
```

	state	location	type	so2	no2	rspm	spm	pm2_5	dat
0	0	Hyderabad	1.0	4.8	17.4	108.833091	220.78348	40.791467	1990 02-0
1	0	Hyderabad	2.0	3.1	7.0	108.833091	220.78348	40.791467	1990 02-0
2	0	Hyderabad	1.0	6.2	28.5	108.833091	220.78348	40.791467	1990

One Hot Encoding

```
dfAndhra=df[(df['state']==0)]
```

```
dfAndhra
```

	state	location	type	so2	no2	rspm	spm	pm2_5
0	0	Hyderabad	1.0	4.8	17.4	108.833091	220.78348	40.791467
1	0	Hyderabad	2.0	3.1	7.0	108.833091	220.78348	40.791467
2	0	Hyderabad	1.0	6.2	28.5	108.833091	220.78348	40.791467
3	0	Hyderabad	1.0	6.3	14.7	108.833091	220.78348	40.791467
4	0	Hyderabad	2.0	4.7	7.5	108.833091	220.78348	40.791467
...	...	...	...	...	...	...	...	...
26363	0	Rajahmundry	2.0	7.0	13.0	71.000000	220.78348	40.791467

```
dfAndhra['location'].value_counts()
```

```
Hyderabad         7764
Visakhapatnam     7108
Vijayawada        2093
Chittoor          1003
Tirupati           986
Kurnool            857
Patancheru         698
Guntur             629
Nalgonda           618
Ramagundam         554
Nellore            408
Khammam            385
Warangal           336
Ananthapur         324
Ongole             317
Kadapa             316
Srikakulam         315
Rajahmundry        311
Eluru              300
Vishakhapatnam     297
Kakinada           288
```

```
Vizianagaram       282
Sangareddy          85
Karimnagar          67
Nizamabad           27
Name: location, dtype: int64
```

```
from sklearn.preprocessing import OneHotEncoder
onehotencoder=OneHotEncoder(sparse=False,handle_unknown='error',drop='first')
```

```
pd.DataFrame(onehotencoder.fit_transform(dfAndhra[["location"]]))
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26363	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
26364	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
26365	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
26366	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

14m 13s completed at 3:24 PM